Density Estimation of Nonequilibrium dynamics of the Ising Model Using Generative Neural Networks

Samuel D. Gelman and Guy Cohen*

School of Chemistry, Tel Aviv University, Tel Aviv 69978, Israel

(Dated: January 23, 2023)

Entropy estimation is a key challenge in the field of statistical mechanics. Systems of meaningful size require methods that can yield the entropy despite having an intractably large phase space In this work a generative model, PixelCNN, was used to create a tractable distribution modeled after the Ising model in and out of equilibrium. Using this model for density estimation proved to outperform existing methods both in accuracy and scaling, in at least some cases. he method is applicable to any system that can be well represented by pixelated data. Its usefulness was demonstrated by exploring the role of entropy in non-equilibrium Glauber dynamics, describing the behavior of an Ising model driven by a rapidly oscillating magnetic field.

* gcohen@tau.ac.il

CONTENTS

I. Brief history of Entropy 3

1. Thermodynamic formulation 3

2. Statistical Mechanics and further abstraction 4

II. Entropy Estimation 6

A. The Difficulties 6

B. Past work 7

III. Continuous Probability Distributions 8

A. Distributions of increasing complexity 8

1. Increasing complexity 8

2. Normal Distributions as Reference 8

3. One Dimensional Normal Distribution 9

4. Rotated Two Dimensional Gaussian 9

5. Radially Symmetric Distributions 10

6. Distribution of Symmetric Wells 10

B. Sampling: The Metropolis Hastings Monte Carlo Algorithm 11

IV. The Ising Model: Equilibrium 11

A. Ising introduction 11

B. Onsager's analytic solution at the limit of an infinite sized lattice 13

C. 14

D. Ising Sample Generation 14

E. Correlation 16

V. The Ising Model: Nonequilibrium 18

A. Glauber Dynamics 18

B. Nonequilibrium Glauber Dynamics 18

VI. Continuous Space Density Estimation 19

A. Increasing Complexity 19

B. Factorization: Entropy and Independence 20

C. Linear Correlation 20

D. Principle Component Analysis 21

E. Kernel Density Estimation 22

VII. Ising Model Density Estimation 24

A. Lossless Compression Algorithms 24

B. Autoregressive models 25

C. Applications towards Entropy Estimation 25

D. PixelCNN 26

E. Training and validation loss 28

F. Optimizing Hyperparameters 30

VIII. Continuous Space Density Estimation 31

A. 1-Dimensional Density Estimation using histogram 31

B. Combining PCA with histogram density estimation 32

C. Tricking Principle Component Analysis 34

D. Overcoming limited sample size 35

IX. Working on higher dimensional objects 36

A. Analytic bench-marking of PixelCNN 36

B. Exploring non-equilibrium phase space 37

C. Exploring the chaotic regime 40

References 44

Introduction:

I. BRIEF HISTORY OF ENTROPY

1. Thermodynamic formulation

Thermodynamics is the branch of science that investigates the states of matter. The foundations of thermodynamics were built on experimental observations and were used to formalize phenomena like heat, work and temperature. Early work included that of Sadi Carnot as a means to improve the efficiency of steam engines [1] and was later formalized by Lord Kelvin as the study of "thermodynamics" or the movement of heat between contiguous bodies

The bedrock of thermodynamics is built upon a few fundamental principles known as the laws of thermodynamics. The first law is the conservation of energy and it outlines a range of possible thermodynamic behaviors of a system. An isolated system, in a given state, with internal energy U_1, can only move to a new state with equal internal energy, such that $\Delta U = 0$. An understanding which relies just on this first law would allow for a multitude of reversible processes of the system to take place. The observed reality is far different. For any given set of parameters, for an isolated system, there exists a well-defined state that the system will spontaneously and irreversibly evolve towards, this state is known as the equilibrium state [2]. To explain this phenomena the second law of thermodynamics is needed.

Rudolf Clausius made the observation that heat cannot spontaneously pass from a colder to a hotter object without some other energy transfer happening at the same time [3]. This statement came to be recognized as one of the early statements reflecting the second law of thermodynamics. After over a decade of exploring the phenomena of interacting bodies and their transfer of heat, Rudolf Clausius coined the term entropy in 1865 to describe the energy lost as heat from any irreversible process [4].

Entropy was defined as being a state function which is additive for composite systems and is maximized at equilibrium. Thermodynamic relationships between entropy and the other thermodynamic state functions, namely internal energy and volume, were formulated throughout the 19th century by the likes of Herman Van Helmholtz and Josiah Willard Gibbs [5] [6].

2. Statistical Mechanics and further abstraction

Ludwig Boltzmann drew a connecting thread between the macroscopic state functions, namely entropy, to the total number of microscopic states available to the system. He did this succinctly and beautifully and his equation for entropy,

$$S = k_B \log(\mathit{\Omega}), \tag{1}$$

appears on his tombstone in Vienna. Here Ω represents the total number of microstates available to the system and k_B is Boltzmann's constant, which relates the average kinetic energy of a particle in a gas to the overall temperature of the gas. Boltzmann defines the degree of disorder and uses

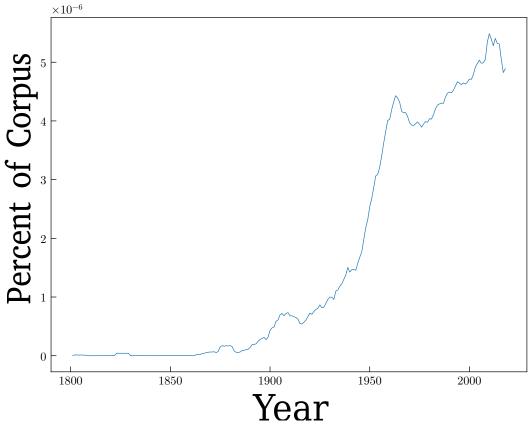

Figure 1. This plot shows the use of the word entropy over the last 218 years. A jump of 2.35x is seen after Shannon's landmark publication in 1948. The data is taken from the Google's Books Ngram project. The project was the result of cataloging the use of words and phrases from a vast corpus of digitized literature spanning several hundred years [7].

that to define the entropy of a system. This language is now commonplace when trying to describe the concept of entropy.

Boltzmann anchored the thermodynamic definitions of energy and temperature to purely statistical observations of the system. But this statistical definition of entropy didn't only manifest itself in the discipline of thermodynamics.

Claude Shannon spawned an entire scientific theory when he published his paper in 1948 titled "A Mathematical Theory of Communication." Here Shannon tries to quantify the amount of information contained in a given observation and lay out how that affects its ability to be communicated. He defines his conception of informational entropy with a formula extremely similar to that of Boltzmann's famous theorem:

$$S = -\mathbf{K} \sum_{i=1}^{k} p\left(i\right) \log\left(p\left(i\right)\right), \tag{2}$$

K is a positive constant which can be used as a normalization term, and p is the probability of event i happening. S is replacing the variable H used in the original paper to maintain consistency through this work.

The abstraction and applicability of the idea of entropy caused it to spread through multiple disciplines, which is reflected by an increased adoption of the word after Shannon's formulation as seen in figure 1. Entropy entered the minds of economists, sociologists, statisticians and computer scientists [8] [9] [10] [11].

II. ENTROPY ESTIMATION

A. The Difficulties

When trying to learn about a system through experiments or simulations it is likely that the amount of observational data available will only make up a small portion of the total available sample space present. Often it won't be known just how big that available space is. In practice, samples $\{X_i\}$ can be obtained from a distribution $P(X) = \frac{p(X)}{Z}$, where that distribution is normalized by some value Z, referred to as the partition function, which is not necessarily known. Without knowing Z, $p(X)$ gives the unnormalized distribution of $P(X)$. Through this limited vantage into the configuration space, some values can be estimated, for example the internal energy U:

$$U = \int \mathrm{d}X \, P(X) \, E(X) = \frac{1}{N} \sum_i E(X_i) + O\left(\frac{1}{\sqrt{N}}\right). \tag{3}$$

Here X is the random variable describing a configuration of the system and $E(X)$ is the energy of the system at that configuration. X_i is the ith sample, with N samples taken in total. These expectation values approach the true value with an error of $O\left(\frac{1}{\sqrt{N}}\right)$, allowing relatively good estimations to b obtained independent [12]. of the size of the system [13]. There are many uses to defining these properties but there are limits to what one can learn from them about the system as a whole.

For certain system properties the full distribution must be known. Calculating entropy, S, requires $P(X)$ to be known explicitly:

$$S = \int \mathrm{d}X \, P(X) \ln(P(X)) = \frac{1}{N} \sum_i \ln(P(X_i)) + O\left(\frac{1}{\sqrt{N}}\right). \tag{4}$$

Analytically calculating the entropy for systems of meaningful sizes grows intractable quickly [14], solving for the partition function Z simply becomes too difficult. In order to calculate the analytical entropy, the probability density function must be known and an integral must be taken over all possible states.

B. Past work

In thermal equilibrium, energy fully determines the probability of occupying a configuration and several powerful, generic techniques have been developed to take advantage of this. Despite this fact, algorithms like Metropolis-Hastings and importance sampling can only manage to efficiently sample relatively low dimensional distributions [15]. Methods to better and more quickly sample this large space do exist but often require many Monte-Carlo simulations and are also limited in scalability. These approaches become ineffective when the space of exploration reaches a meaningful size because of how the configuration space scales with system size [16].

Even when brute force sampling could theoretically approach a valid estimation for the density - certain configurations could be difficult to reach using stochastic sampling methods. To overcome this barrier the multihistorgram method was developed which connects many separate yet overlapping distributions and stitches them together. Later improvements to importance sampling managed to overcome that limitation and succeed similar results in just one run [17],[14].

The broad histogram approach leverages this connection by taking random walks through energy space while constructing a distribution which approaches that of the density of states [18]. This method has been built upon to make substantial progress in the field [19] [20] [21] [22] [23]. Yet the method is restricted to distributions in equilibrium.

Estimating the entropy by making small changes to the analytic solution has also seen success [24, 25] [26] [27]. Work has been done to extend this method to non-equilibrium states but remains restricted in reach [28]. However, for non-equilibrium systems, even when simulation is possible, the probability density generally cannot be expressed in terms of a single, known variable [29], [30].

Most approaches therefore resort to enumerating the configuration space [31], for example by coincidence counting. This method quickly runs into issues for large systems [32].

Another more recent method measured the degree of compressibility to measure the entropy [33] by using lossless compression algorithms like zip and Lempel-Ziv compression.

There are multiple approaches to minimize the mutual information between increasingly large subsystems using artificial neural networks [34] [35] [36].

Models

III. CONTINUOUS PROBABILITY DISTRIBUTIONS

A. Distributions of increasing complexity

1. Increasing complexity

The probability space of a distribution increases exponentially with the number of random variables being considered. This is because each variable adds an additional dimension to be measured along, which can be thought of as adding an another axis in the probability space. This causes an increase in the volume of the space, and volume scale exponentially with dimension.

So even when simulation of a distribution is relatively easy, the amount of observations necessary to make statements regarding the entropy quickly grows intractable as the system size increases. The disparity between the amount of observations needed versus the dimensionality of a system led to the popularization of the phrase “the curse of dimensionality”. It was often used in reference to the fact that statistical techniques that were effective in low dimensions became useless as the dimensionality of the system increased.

2. Normal Distributions as Reference

The Normal or Gaussian distribution is a special distribution in that it describes the normalized statistical behavior of many independent and identically distributed random variables even when those variables themselves are not normally distributed [37].

Normally distributed distributions can be described as:

$$\mathrm{N}_{\mu\sigma}\left(x\right) = Ae^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{5}$$

where μ is a well defined mean, σ^2 is the variance and $A = \frac{1}{\sigma\sqrt{2\pi}}$. The normal distribution is one which has a unique relationship relating the mean and standard deviation when taking the expectation of the distribution, namely:

$$\left\langle (x-\mu)^2 \right\rangle = \sigma^2. \tag{6}$$

This relationship makes the entropy of normal distribution extremely simple to calculate analytically. The differential entropy can be written in terms of the expectation as:

$$S\left(x\right)=\int_{x}f\left(x\right)\log f\left(x\right)dx=\left\langle\left[-\log\left(f\left(x\right)\right)\right]\right\rangle. \tag{7}$$

Utilizing this relationship, the analytical solution for the differential entropy can be derived as follows:

$$S\left(x\right)=\left\langle\left[-\log\left(\mathrm{N}_{\mu\sigma}\left(x\right)\right)\right]\right\rangle,$$

$$=-\left\langle\left[\log\left(\left(2\pi\sigma^{2}\right)^{-\frac{1}{2}}e^{\left(-\frac{1}{2\sigma^{2}}(x-\mu)^{2}\right)}\right)\right]\right\rangle,$$

$$=\frac{1}{2}\log\left(2\pi\sigma^{2}\right)+\frac{1}{2\sigma^{2}}\left\langle\left[(x-\mu)^{2}\right]\right\rangle,$$

$$=\frac{1}{2}\left(\log\left(2\pi\sigma^{2}\right)+1\right). \tag{8}$$

Using this analytical benchmark will allow for a standard with which to test various methods [38].

Another powerful propriety of the symmetric Gaussian is that it can be factorized effectively . The dimensionality reduction of Gaussian distributions is

3. One Dimensional Normal Distribution

The one dimensional normal distribution, figure 2 panel (a), is a simple example of a distribution that can be easily approximated using brute force methods like histogram binning. The fact that the analytic solution is known makes it an intuitive example where simplistic methods can be effective.

4. Rotated Two Dimensional Gaussian

The rotated, two dimensional normal distribution, figure 2 panel (b), is convenient because it is known that there is a transformation which can be done to remove the linear correlation between the two dimensions of the distribution. An attempt to reduce the problem to a product of two one dimensional Gaussian distributions will fail using the presented basis vectors but will be successful after the proper transformation.

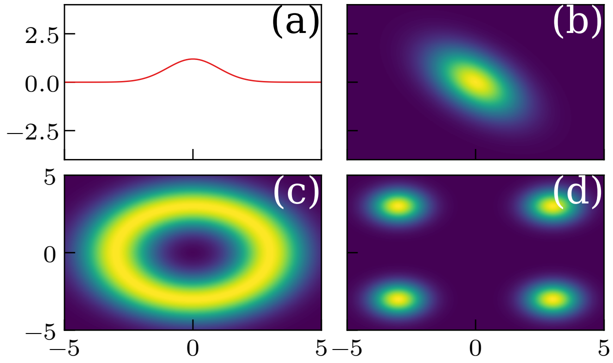

Figure 2. Examples of the continuous space functions used to explore the applicability and limitations of various density estimation and dimensional reduction techniques. Panel (a) is a simple, one dimensional normal distribution with mean 0 and standard deviation 1. Panel (b) is a two dimensional normal distribution where the two random variables are linearly correlated. Panel (c) is an example of a distribution with radial symmetry. Panel (d) is a distribution which has peaks separated from one an other and symmetrically distributed in space.

5. Radially Symmetric Distributions

The radially symmetric distribution figure 2 panel (c), is similar to the rotated Gaussian it cannot be factorized in one coordinate system while it can be in another, in this case within a radial coordinate system. However, contrary to the above distribution, no linear transformation can factorize it.

6. Distribution of Symmetric Wells

The final toy model used, figure 2 panel (d), is a case where the distribution is lacking a clear direction of correlation because of its symmetry. While having a radial symmetry is also has a certain degree of linear correlation. Because of this, it can dangerously provide false positives using the methods applied to the rotated Gaussian. For this reason it is used to show the effectiveness of other density estimation techniques.

B. Sampling: The Metropolis Hastings Monte Carlo Algorithm

As mentioned above, given a distribution $P(X) = \frac{p(X)}{Z}$, without knowing Z, samples, $\{X_i\}$, can still be drawn which accurately reflect the distribution. This is often the case for experiments done in the lab. Extending this ability to simulations was of optimal importance to increase the usefulness of computers as a tool for scientific investigation by properly leveraging the strength which they could have as tools for investigating phenomena and building accurate models.

In 1953 Nicholas Metropolis developed an algorithm that did just this for symmetrical distributions [39]. W.K. Hastings extended it to more general cases in 1970 [40].

The algorithm works by taking a random walk within a configuration space where candidate steps are randomly chosen based off predetermined criteria. The probability of accepting a candidate step is more likely when moving from lower values of $p(X)$ to higher ones.

More specifically the algorithm starts with an initialization sample X. Then it proposes a new sample, X', which had been generated by taking a step as described above. A random number, $a \in [0, 1]$ is generated from a uniform distribution and compared to an acceptance function which also outputs a single real number between 0 and 1:

$$Q(X, X') = \min\left\{\frac{p(X')}{p(X)}, 1\right\}. \tag{9}$$

If $Q(X, X') \geq a$ then X' is accepted and the procedure repeats, comparing a new candidate step to the previously accepted X'. In the event that a is larger than the output of the acceptance function, X' is reject4ed and a new candidate step is measured against our original X. All or some of the samples are stored and make up a data set proportional to the true population. The method can be extended to any dimension, though at high dimensions modifications are required to insure accurate sampling [41].

IV. THE ISING MODEL: EQUILIBRIUM

A. Ising introduction

The two dimensional Ising model is constructed of sites, i, which can take on one of several values. Here the sites take on values of $\alpha \in [-1, 1]$. The Ising model is often referred to as a spin model, and the positive and negative values at a site can be thought of as an up or down spin, respectively.

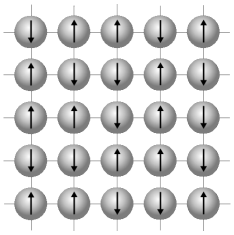

Figure 3. Shows a spin representation of an Ising model, where cells in the grid assume a binary value and are arranged in a spatially significant lattice structure [42].

The model can be constructed in any dimension, but is very often, as well as in the case of this work, considered in two dimensions. Sites are arranged in a grid which is usually a square lattice. An illustration can be found in figure 3.

Any given configuration has a well defined energy as described by its Hamiltonian:

$$E\left(\alpha\right) = -\sum_{(i,j)} J_{i,j} \alpha_i \alpha_j - H \sum_j \alpha_j. \tag{10}$$

The model described in equation (10) is a two dimensional lattice structure of an Ising model. The Hamiltonian, E, takes as an input α, which here is a spin configuration which has an index $\langle i, j \rangle$ denoting sites. J is a the coupling constant that represents the interaction strength between neighboring spins. When $J > 0$, which it will be taken to be for the scope of this work, and neighboring spins are of opposite values, the interaction energy will be positive, contributing energy to the system. The opposite is true for identical spins, meaning the model energetically favors uniformity of spins across the model. The second term in the Hamiltonian acts on sites individually and represents an external magnetic field. The magnitude of the field is H.

The Boltzmann distribution is used to find the configuration probability in equilibrium:

$$P_\beta\left(\alpha\right) = \frac{e^{-\beta E(\alpha)}}{Z_\beta}. \tag{11}$$

Here $\beta = \frac{1}{k_B T}$ and k_B is Boltzmann's constant .

There exists a critical temperature where the system undergoes a phase transition. This is the point where the average mean magnetization of the system goes to zero. This point is not simply a construct within theoretical models. The phenomena was first observed and recorded by Pierre Curie just before the turn of the 19th century [43]. He observed that the ferromagnetic properties of iron were lost at a high temperatures and proceeded to measure a catalog this temperature for many metals. This point is known as the Curie temperature and can be seen in figure 4.

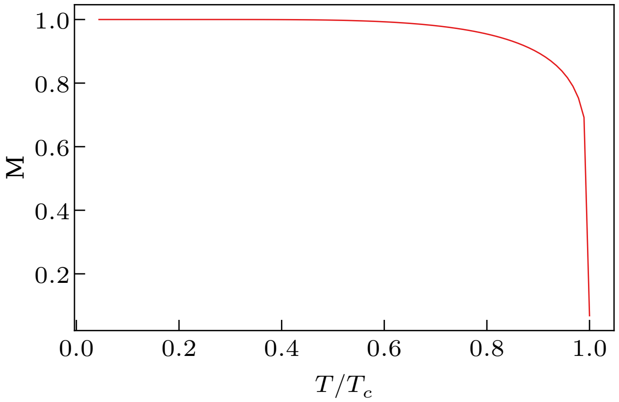

Figure 4. Shows the mean magnetization of the square Ising model as a function of the temperature. The temperature here is normalized by the curie temperature, T_c, such that 1 on the y-axis marks the critical point of phase transition. The figure was generated using a formula derived from Onsager's solution where $M = \left[1 - \sinh^{-4}(2\beta J)\right]^{1/8}$ where $\beta = \frac{1}{T}$for all values $T < T_c$ [44].

The presence of a Curie temperature in the Ising model as well as in solid metals begins to demonstrate the usefulness of the model and starts to explain its wide ranging use and popularity. This story is more beautifully closed because of the finding made a little over two decades after the Ising model was introduced to the scientific community: an analytic solution.

B. Onsager's analytic solution at the limit of an infinite sized lattice

Lars Onsager, in 1944, found an analytic description of the 2d Ising model [44]. Using his formulation, the entropy is known as a function of the temperature and is available as an analytic benchmark for entropy estimation techniques.

The analytic solution for the entropy of a 2d-lattice in the case where there is no external magnetic field is as follows:

$$-\beta F = ln(2) + \frac{1}{8\pi^2}\int_0^{2\pi} \mathrm{d}\theta_1 \int_0^{2\pi} \mathrm{d}\theta_2 \ln[\cosh{(2\beta J)}^2 - sinh(2\beta J)(cos(\theta_1) + cos(\theta_2))], \quad (12)$$

$$k = \sqrt{1 - \frac{1}{\sinh{(2\beta)}^2}},$$

$$U = \frac{-J}{\tanh{(2\beta J)}} \cdot \left(1 + \frac{2}{\pi}\left(2\tanh{(2\beta J)}^2 - 1\right) \cdot \int_0^{\frac{\pi}{2}} \frac{1}{\sqrt{1 - \frac{4k}{(1+k)^2 \sin(\theta)^2}}}\right). \quad (13)$$

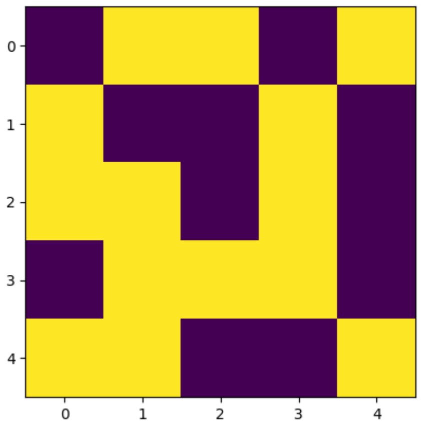

Figure 5. An example of a generated Ising sample with side length $L = 5$ where yellow represents one of the spin states and the purple represents the opposite one.

From the free energy, F, and the internal energy, U, the entropy can be found by manipulating the expression for the Helmholtz free energy [2]:

$$S = \beta \left(U - F \right).$$

This describes the isotropic case, where horizontal and vertical nearest neighbor interactions are equal at the thermodynamic limit. Although they are similar, the system has different properties at finite size [45]. Correcting for the finite size limit is necessary to properly assess the effectiveness of any approach working with sample data. Methods have been developed to correct for the finite size effects on the entropy [46]. These approximation are used as the analytic benchmarks for the results and figures which follow in this work.

C.

D. Ising Sample Generation

The fact that the Hamiltonian is diagonal in the spins makes classical Monte Carlo for the Ising model possible. For example, samples can be obtained using the Metropolis-Hastings algorithm, which is described in further detail in the Methods section.

The relative probability of finding the Ising model in one state as opposed to another is clearly outlined by the model's Hamiltonian, equation (10).

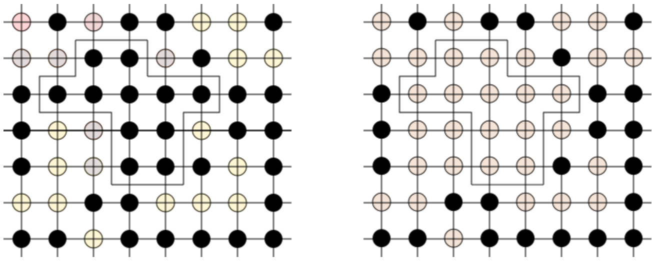

Figure 6. This is a representation of an example cluster using the Wolff algorithm. These clusters are easily defined for low temperatures and allows for sampling similarly likely configurations even if they as spatially distant from one another.

A reliable set of samples can be generated by taking snapshots of a time evolution of the system [39]. Here, "reliable" means a relatively small but representative fraction of the total population. In order to insure that samples are adequately uncorrelated, a sufficiently large number of Monte Carlo steps should be taken between saving samples. These nearly uncorrelated samples represent the thermal distribution and can be obtained with modest demands on computer time and power. For this work, the correlation of the samples were tested to ensure adequate waiting times and is described below in more detail.

For samples at high temperatures the Metropolis-Hastings algorithm was used. For low temperature samples, the Wolff algorithm which produces low correlated samples much more efficiently, was used [47]. The latter algorithm works by leveraging the long distance correlations which become increasingly more dominant at low temperatures when the distribution is more strongly influenced by the relative energy of the configurations. This tendency causes algorithms like Metropolis-Hastings, with only local changes between configurations, to get caught in local minima.

The Wolff Algorithm also works by stochastically exploring configuration space. Unlike the Metropolis-Hastings algorithm which explores possible configurations by making incremental changes to a single random variable at a time, the Wolff algorithm creates clusters and flips those entire clusters with a probability of 1.

The algorithm checks all possible links of a site to test whether or not the cluster will extend over that link. Links are accepted to become part of the enlarging cluster with a probability of

$$p_+ = 1 - e^{-2\beta\sigma_-\sigma_+}, \tag{14}$$

where $\sigma_-$and σ_+ are the spin values at either end of a given link. When spins are opposite, it is impossible for the link to be accepted in the cluster. For high temperatures, when β is low, the likelihood of forming large clusters also drops significantly.

When a cluster is being formed, each available link is only tried once. That said, a site can be included into the cluster via any of its links so even if it fails once it might still become included via the link of a different neighboring site.

This algorithm allows for a freer movement though the configuration space when long ranged correlations are present. That said, it is a more computationally expensive algorithm to run so it is only used for sample generation when necessary.

E. Correlation

Metropolis and Wolff Monte-Carlo sample generation works by making incremental changes to the current configuration, comparing the relative energies, and then probabilistically deciding to accept or reject a sample using those energies as a metric. When building the data sets for this paper it was important that the samples chosen to be accepted from the Monte Carlo algorithm were uncorrelated. Given the incremental nature of the sampling algorithm, samples can easily be correlated one to the next if a sufficient number of Monte Carlo steps is not waited between accepting a sample into the data set. It is therefore important to have a metric for testing the level of correlation found between samples.

Samples, X, can be generated concurrently using several different computational cores where each core is running a separate trajectory of the Metropolis-Hastings or Wolff algorithm. The data can then be organized using three indices,

$$X_d^{n,i}, \tag{15}$$

where $d \in \{1, \ldots, D\}$ denotes a trajectory, and is the total number of trajectories; $n \in \{1, \ldots, N\}$ denotes a sample, and is the total number of samples in each trajectory; and $i \in \{1, \ldots, I\}$ and I is the total number of pixels in each sample.

Given this structure, the mean value for an individual pixel can be calculated across trajectories as:

$$\left\langle X^{n,i} \right\rangle = \frac{1}{D} \sum_{d=1}^{D} X_d^{n,i}. \tag{16}$$

This mean is then calculated for each value in each sample. The term for the variance of our pixel values through all trajectories is then:

$$\operatorname{var}(n,i) = \frac{1}{D} \sum_{d=1}^{D} \left(X_d^{n,i} - \left\langle X^{n,i} \right\rangle \right)^2. \tag{17}$$

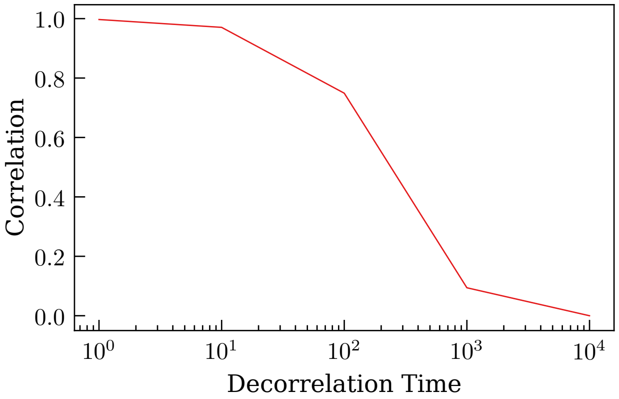

Figure 7. A trajectory was taken on the $L = 64$ lattice Ising model parameterized with T=3.5, where the decorrelation time corresponds to the number of Monte Carlo steps waited between taking samples. The correlation values are plotted as a function of those waiting times. A value of 1 means that collected samples are fully correlated while a value close to 0 means they are uncorrelated. The figure shows that decorrelation times on the order L^2 are needed before samples can be confidently decorrelated .

Given this variance between trajectories, a general function for the covariance for any pixel with any other pixel in a given trajectory can be written as follows:

$$\mathrm{cov}\left(n,i;n',i'\right) = \frac{1}{D}\sum_{d=1}^{D}\left[\left(X_d^{n,i} - \left\langle X^{n,i}\right\rangle\right)\left(X_d^{n',i'} - \left\langle X^{n',i'}\right\rangle\right)\right]. \tag{18}$$

For the purposes of this work, it is important to know how correlated consequent values of spins are to one another. This is because they have the highest "risk" of being sampled prematurely. To systematically measure these correlation the following covariance values are important:

$$\mathrm{cov}\left(n,i;n+1,i\right) = \frac{1}{D}\sum_{d=1}^{D}\left[\left(X_d^{n,i} - \left\langle X^{n,i}\right\rangle\right)\left(X_d^{n+1,i} - \left\langle X^{n+1,i}\right\rangle\right)\right]. \tag{19}$$

Given the above, the degree of correlation in the data can be defined as follows:

$$\mathrm{corr}\left(X\right) = \frac{1}{I}\frac{1}{\left(N-1\right)}\sum_{i=1}^{I}\sum_{n=1}^{N-1}\frac{\mathrm{cov}\left(i,n;i,n+1\right)}{\sqrt{\left(\mathrm{var}\left(i,n\right) * \mathrm{var}\left(i,n+1\right)\right)}}.$$

Ising data generated by Monte Carlo algorithm was taken during several runs parameterized by the decorrelation time, or the number of Monte Carlo steps taken between each recorded data point. Figure 7 shows the correlation as a function of the decorrelation time.

V. THE ISING MODEL: NONEQUILIBRIUM

A. Glauber Dynamics

The Ising model is by nature probabilistic. To consider dynamical properties, one needs to explicitly define a sequence of configurations and a time step. An intuitive and common definition for a single time step in simulated dynamics for the square Ising model is L^2 Monte Carlo attempts at flipping a site, where L is the side length and L^2 is the number of sites [48]. Saving samples and marking them with an evolving time signature allows one to observe dynamics both in and out of equilibrium.

The algorithm is carried out in the following way:

- Start from some initial configuration with L^2 spins.
- For every time step, repeat the following L^2 times:
  - Pick a random spin and compute the energy difference ΔE for flipping it.
  - Flip the spin with probability $\frac{e^{-\beta\Delta E}}{1+e^{-\beta\Delta E}}$ (i.e. enforce detailed balance).

ΔE and β are the same as those outlined above.

By defining a time step as L^2 Monte Carlo steps, the dynamics can be mapped as frame by frame snapshots, taken one time step apart from the next.

B. Nonequilibrium Glauber Dynamics

With an unambiguous definition for time in which to portray dynamics, equilibrium and nonequilibrium dynamics can be studied. This is true even though the Hamiltonian describing the Ising model, and the density of samples, are defined for in equilibrium conditions. The reason for this is that at any given time step, the effective conditions acting on the sample can be taken as equilibrium conditions.

The Hamiltonian of the Ising model described in equation (10) has a term h which adds a spin bias. This bias can be viewed as an effective magnetic field. This magnetic field, h, can be parameterized by a time variable defined by Glauber dynamics t as follows:

$$h\left(t\right) = H\sin\left(\omega t\right), \tag{20}$$

where H is the magnitude of the magnetic field and ω defines the frequency.

Equation (20) along with the temperature gives a rich parameter space to explore dynamics. Additionally, given the cyclic nature of the the oscillating magnetic field, each oscillation can be view as a self-contained experiment. This supplies extremely inexpensive numerical experiments for investigation. The fruits of this exploration will be shown in the results section.

Methods

VI. CONTINUOUS SPACE DENSITY ESTIMATION

A. Increasing Complexity

Unfortunately, simple Monte Carlo sampling methods quickly collapse when the dimensionality of the problem increases.

The issues become apparent even when slightly increasing the complexity of the distribution of interest. Moving from 1-d to 2-d will immediately display the approaches shortcomings.

Suppose samples are drawn from a 1-d distribution, $p\left(x\right)$, whose values fall within the continuous range of 0.0 - 10.0. As a means of density estimation, the output space can be broken into discrete bins and as observations are made they can be placed into the appropriate bin. After a satisfactory number of samples have been drawn, these bins can be used as an estimate from which to calculate the entropy. Instead of integrating over the whole space, a sum can be taken over each bin and the entropy can be estimated:

$$S_{hist} = -\sum_{i=1}^{n} p(x_i)\ln(p(x_i))\Delta x, \tag{21}$$

where bin value is $[x_1, ..., x_n]$ and Δx is the size of each bin.

With this framework in mind, suppose 100 samples were taken, that leave give an average of 10 samples per bin. If the distribution of interest was now 2-d, $p\left(x, y\right)$, to reach the same level of resolution 100 bins would now need to be filled. If the same amount of data was used as in the 1-d case, there would only be, on average, one point per bin. Either resolution would need to fall by a factor of almost three in each dimension or 10 times the amount of samples would need to be generated.

Finding ways to circumvent this particular “curse of dimensionality” is crucial for estimating the entropy of more complex systems.

B. Factorization: Entropy and Independence

It turns out that distributions comprised of multiple independent random variables don't pose a problem to the calculation of entropy. This is because the entropy of such distributions can be calculated by taking the sum of the two values of entropy of its component random variables. For a distribution P with independent variables x and y the entropy can be written as:

$$S(x,y) = -\int\int P\left(x,y\right)\ln\left(x,y\right)dxdy$$

$$= -\iint P\left(x,y\right)\ln\left(x\right)dxdy - \iint P\left(x,y\right)\ln\left(y\right)dxdy$$

$$= -\iint P\left(x,y\right)dy\ln\left(x\right)dx - \iint P\left(x,y\right)dx\ln\left(y\right)dy$$

$$= S\left(x\right) + S\left(y\right). \tag{22}$$

This holds true for any number of random variables given they are independent. This property of independence and entropy will act as an important foothold for navigating the problems of entropy calculation in higher dimensions.

C. Linear Correlation

Nature is highly interconnected and systems often comprise many dependent random variables. That said, there exist patterns of dependence which can be utilized to allow for a simplification in the difficult task of calculating entropy. One such pattern is the degree of linear correlation.

A system can be described as linearly correlated when an increase of one variable correlates to a linearly consistent change in a second variable. When an increase in one variable is mirrored by an increase in another then the linear correlation is said to be positive. If, instead, an increase of one variable is met with a decrease in the other than the variables would have a negative linear correlation.

In figure 8 there are two simple examples of data with low, left panel and high, right panel, linear correlations. The entropy of the uncorrelated example can be calculated by summing the entropy of x and y independently. This is not the case for the lineally correlated model.

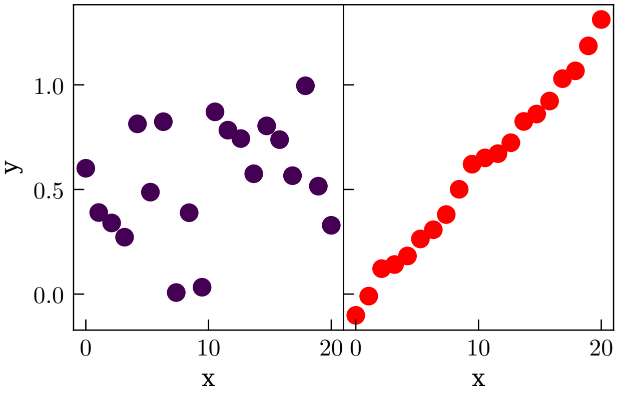

Figure 8. Examples of distributions with low (left) and high (right) correlation plotted on arbitrary axes x and y.

D. Principle Component Analysis

For any linearly correlated distribution there exist a vector which best represents the correlation of its data. We can define such a vector as one which, when data is orthogonally projected onto it, maximizes the variance. It can also be said that it is the vector which minimizes the mean-squared difference between the data and their projected values on to such a vector [49]. This vector is called the first principle component. There is a principle component for every dimension of the sample data, all of which are orthogonal to each other and all of which decrease in magnitude from one to the next.

The principle components can be found by calculating the eigenvectors of the distribution's covariance matrix. Computing a linear transformation in which the data set is multiplied by these eigenvectors will set the principle component vectors as the new basis vectors. This new, transformed, data set will be the most accurate approximation to an uncorrelated distribution.

Figure 9 is what happens after performing a principle component analysis on the linearly correlated data in figure 8. This new, linearly transformed, data set gives an entropy similar to the original when adding the x and y components and costs much less computational time to arrive at that answer. It is worth noting, that while independence implies a lack of correlation, the converse is not true, so this factorization is generally an approximation.

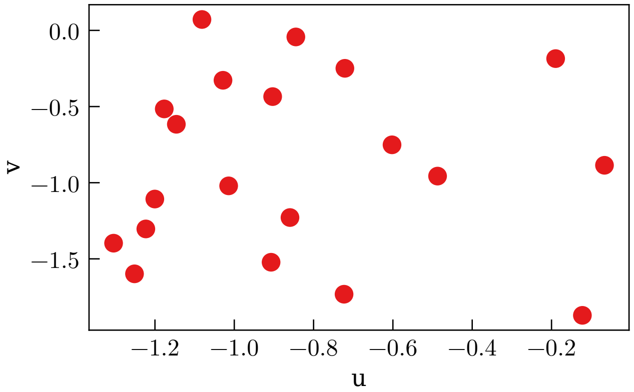

Figure 9. Result of using principle component analysis on the 2d linearly correlated data set in Figure 8 on page 21b. The graph now strongly resembles the uncorrelated data found in 8a

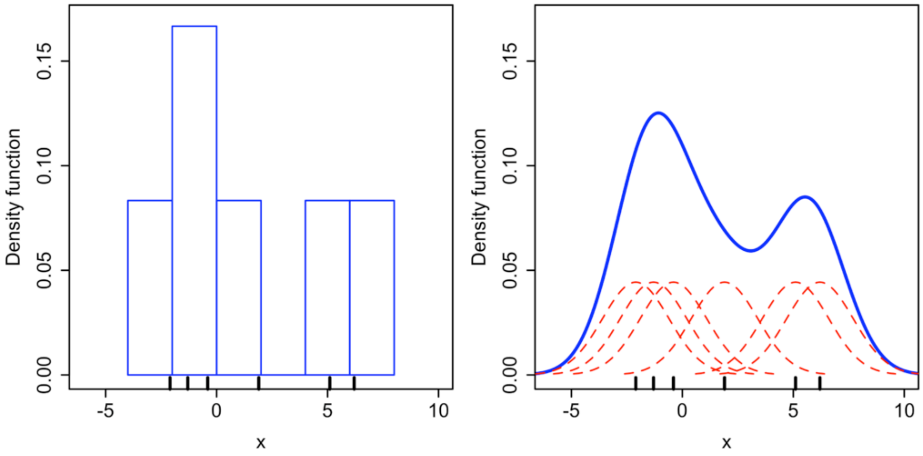

Figure 10. Given six observations of a one dimensional distribution two algorithms are being used to represent the presented data; histogram (left) and kernel density estimation using a normal function as the kernel function (right). The kernels act as a way to buffer the bias of limited data and generate much smoother estimations for the distributions [50].

E. Kernel Density Estimation

Kernel Density Estimation (KDE) is another tool that provides a means of extrapolating density from limited sample observation without assuming that the data is uncorrelated. The kernel, for which the name arrives, is a continuous, non-negative, function where the value of the sample X is a constant in the function. For example,

$$P_{\mathrm{KDE}}\left(X\right) = \frac{1}{n}\sum_{i=1}^{n} K_h\left(X - X_i\right). \tag{23}$$

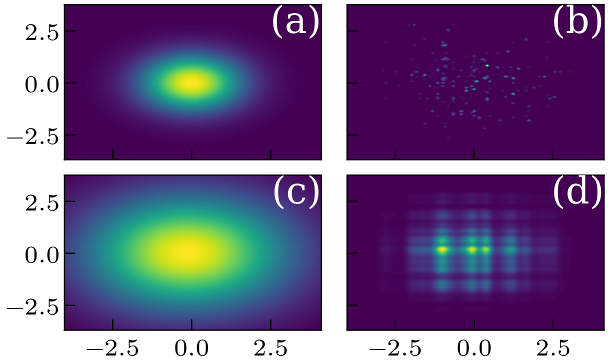

Figure 11. Representation of a Gaussian distribution, subplot(a,1) and one hundred data points placed into a histogram, subplot(a,2). The bottom two subplots represent kernel density estimation using first a bandwidth which is too low in subplot(b,1) and the other which is not low enough in subplot(b,2).

The functions have a smoothing parameter, h, which is often referred to as the bandwidth. The bandwidth parameter is tuned so that the minimum value can be found to accurately represent the distribution. When tuning the bandwidth there is a trade off between bias and variance. These functions can assume a wide range of values and the dimensionality of them can vary with the needs of the estimation task. A common kernel is the Gaussian kernel,

$$K = e^{\frac{-(x_i - x)^2}{2h}}, \tag{24}$$

where the sample, x_i, represents the well-defined mean of the function and the bandwidth, h, is the standard deviation.

It is common to parameterize the bandwidth by the standard deviation of the data set. This gives sharper functions when the data has lower deviations and broader functions when deviations are larger.

In figure 11 a Gaussian distribution is shown as well as a collection of samples drawn from it. Using that sample data, a KDE was used to generate two distributions. In one the bandwidth parameter was set too low, this created a model where the standard deviation is much larger than that of the actual distribution. In the second example, the bandwidth was not set low enough. The bias ended up being much too strong in this case and the distribution barely even resemble a Gaussian. In theory, the KDE should be able to build an extremely good model given that the kernel is the same function as that which was trying to be represented.

To calculate the entropy from this model distribution generated by KDE,

$$S = \int P\left(x\right) \ln\left(P\left(x\right)\right) dx \simeq \frac{1}{N} \sum_{i=1}^{N} \ln P\left(x_i\right) \simeq \frac{1}{N} \sum_{i=1}^{N} \ln P_{KDE}\left(x_i\right),$$

where $P\left(x\right)$ is the true distribution. To lower stochastic noise, one should re-sample the distribution between the construction of the P_{KDE} and the estimation of the entropy. This can be done by using a Metropolis algorithm on P_{KDE} or by using the means originally used to get the first set of sample data.

VII. ISING MODEL DENSITY ESTIMATION

In the case of the Ising model, the number of states grows exponentially with an increase in system size at a rate of 2^N where N is the number of sites [16]. This necessitates the use of methods that only need a small fraction of possible samples to represent a much bigger space.

A. Lossless Compression Algorithms

When approaching the task of measuring entropy it can be helpful to view the concept of entropy from all formulations made in the past. Information theory, born out of the work by Shannon [51] and Kolmogorov [52], drew mathematically identical lines between the statistical-mechanics definition of entropy and the informational entropy at the limit of large data. Shannon defined the entropy as the average information gained per observation of a system. This is in a scheme where the means for communicating the information of our system is ideal. In this way the entropy is a measure for the limit of lossless compression.

Lossless compression algorithms work to realize the theoretical concepts set out by Kolmogorov's formulation of complexity.

Lempel and Ziv (LZ) laid out schemes which ended up being the most adopted approach for implementation of lossless-compression algorithms [53], [54] [55].

The algorithms work by receiving data sequentially and processes that input and replacing repeating segments with pointers to past instances of that segment. The data must belong to a finite alphabet. Given enough data it has been shown that the ratio of LZ compressed data to the raw input sequences converge to Shannon's definition of entropy [56][57].

B. Autoregressive models

For any set of independent and equally distributed random variables, the likelihood of an event has no dependence of what happened before it. In this way the probability of two events, A and B, occurring is the same as the product of those two events happening independently. This extends to the following equality:

$$P(A|B) = P(A), P(B|A) = P(B), P(AB) = P(A)P(B). \tag{25}$$

For configurations assumed by the Ising model, samples can be viewed with each site being a random variable. If there were no correlations in the data, between sites, then the probability density of a given configuration would be:

$$P(X) = \prod_{i=1}^{n} P(X_i), \tag{26}$$

where n equals the number of sites.

However, it is clear that there are correlations between Ising model data; low temperature samples are almost completely correlated as the model assumes a fully ferromagnetic state.

It is intuitive to look at time as an axis on which to find correlations. This is especially true for natural processes which are known to evolve over time. Statistical formulations on how to implement this approach go as far back as the 1920's with the introduction of the first autoregressive models [58]. These models are feed-forward models in which the density of later observations is constructed as conditionals from the earlier observations. For long strings of events, events that appear later will be dependent on a lot of data which came before it. This rich source of information is the basis for how the PixelCNN machine learning model parameterizes its networks and works as an effective density estimator.

C. Applications towards Entropy Estimation

Like the factorization and KDE approaches, but unlike compression and the measurement of mutual information, the method proposed by this work attempts to estimate the entropy by first performing a different, seemingly harder task: density estimation. With the probability density and samples from a simulation available, evaluation of the entropy becomes trivial.

In particular, we applied PixelCNN++ to the Ising model on a 2D square lattice. We showed that the approach performs well at a variety of system sizes, then demonstrated its usefulness by exploring the role of entropy in non-equilibrium Glauber dynamics describing the behavior of an Ising model within a rapidly oscillating magnetic field.

D. PixelCNN

PixelCNN [59] [60] works by leveraging the chain rule in order to decompose the likelihood of a sample **x** into a product of 1-d distributions where each pixel is its own neural network. Each sample is broken down into its composite pixels and the model is trained by defining the density of the i-th site as

$$p\left(x\right)=\prod_{i=1}^{n} p\left(x_i|x_1,\ldots,x_{i-1}\right). \tag{27}$$

Using this chain of conditional probabilities, PixelCNN can define a likelihood function to act on each pixel and train it against our training data. For every site we can construct a distribution which is dependent on the sites which came before it. These joint probabilities are then maximized to the log-likelihood that a pixel will resemble our training data and formulated as regressive functions parameterized over the pixel values which came before it.

Decorrelated Monte Carlo samples generated from the Metropolis-Hastings algorithm act as the ground truth in the training process. This, in effect, breaks down the complex problem of finding the full distribution into many different classification problems, site by site, where each step is built from the conditional probabilities of steps which came before it. The result is a fully tractable probability distribution which can estimate the entire sample space.

The auto-regressive model is one that creates functions based solely on the actual distribution, using the chain rule of joint probability distributions. A three dimensional vector will provide a simple example with which to articulate this exactly:

$$\boldsymbol{x} = x_1, x_2, x_3; D = 3,$$

$$P_m\left(\boldsymbol{x}\right) = P\left(x_1\right)P\left(x_2|x_1\right)P\left(x_3|x_1,x_2\right). \tag{28}$$

We can write all of these probabilities as parameterized functions which can essentially be linear equations parameterized by weights and biases. These functions can be the input to some other type of function which will restrict its value to something between 0,1.

$$Sigmoid: f\left(x\right) = \frac{e^{x}}{e^{x}+1}, \tag{29}$$

$$P\left(x_1\right) = P_1 = f\left(b_1^1\right) = F_1\left(\theta_1\right), \tag{30}$$

$$P\left(x_2\right) = P_2\left(x_1\right) = f\left(W_1^2 x_1 + b_1^2\right) = F_2\left(x_1; \bar{\theta}_2\right), \tag{31}$$

$$P\left(x_3\right) = P_3\left(x_1, x_2\right) = f\left(\sum_{i=1}^{2} W_i^2 x_i + b_2^2\right) = F_3\left(x_1, x_2; \bar{\theta}_3\right), \tag{32}$$

Equation (28) has the composition of a product and can therefore be easily represented as a sum of logarithms:

$$\log\left(P_m(\boldsymbol{x})\right) = \sum_{d=1}^{D} \log\left(F_d\right). \tag{33}$$

This provides a parameterized equation for the actual distribution. The next step is to define a loss function to train a model on. A convenient function is the Kullbeck-Leibler Divergence [9] [24] because it is a measure of a distance between two distributions and because that distance is zero when the two distributions are the same, thus making it obvious that minimizing the function will give us usable values for our weights. It is written in the form:

$$D_{\text{KL}}\left(P \| Q\right) = \sum_{x \epsilon X} P\left(x\right) \log\left(\frac{P(x)}{Q(x)}\right). \tag{34}$$

In order to utilize the KL Divergence it is necessary to have a reference to the ground truth with which to compare the distribution:

$$P_{\text{real}}\left(x_1, x_2, x_3\right) \rightarrow \left(x_1^s, x_2^s, x_3^s\right); \left[s = 1, .., N\right]. \tag{35}$$

Sample data taken from a distribution is often what is used, acting as an incomplete but useful reference. A delta function is created which is often referred to as the prior distribution and is defined like:

$$P_{\text{prior}}\left(x_1, x_2, x_3\right) = \frac{1}{N} \sum_{s} \delta\left(\boldsymbol{x} - \boldsymbol{x}_s\right). \tag{36}$$

The value of the function is $\frac{1}{N}$ wherever we have an observed sample.

Using this as a reference point a comparison to the models distribution to the prior,

$$D_{\mathrm{KL}}\left(P_{\mathrm{prior}}||P_{\mathrm{m}}\right) = \int P_{\mathrm{prior}}\left(\overrightarrow{x}\right)\log\left(\frac{P_{\mathrm{prior}}(\boldsymbol{x})}{P_{\mathrm{m}}\left(\boldsymbol{x}\right)}\right)d\boldsymbol{x}. \tag{37}$$

The property of logarithms can be used to split the division term inside the logarithm into two, one of which is independent on P_m and is therefore a constant value. Because the absolute value of the KL Divergence is uninteresting and instead the function just needs to be minimized, that term can be ignored. The $\frac{1}{N}$ term from equation (36) will also be ignored leaving:

$$D_{\mathrm{KL}}\left(P_{\mathrm{prior}}||P_{\mathrm{m}}\right) = -\frac{1}{N_{\mathrm{s}}}\sum_{s}\int\delta\left(\overrightarrow{x} - \overrightarrow{x}_{\mathrm{s}}\right)P\left(\overrightarrow{x}\right) = -\sum_{s}\log\left(P_{\mathrm{m}}\left(\overrightarrow{x}_{s}\right)\right). \tag{38}$$

This design allows makes it convenient to leverage the rule of logarithms to free ourselves from the curse of dimensionality. The terms are now linearly dependent and therefore steps along the loss function and move down it step by step, batch by batch.

The trained PixelCNN is a generative model, or a model of $P\left(X\right)$ that can also generate configurations with respect to $P\left(X\right)$ Germain *et al.* [61]. The ability to generate samples with respect to $P\left(X\right)$ acts as an effective means to quickly asses that the model has learned the general "idea" by comparing generated samples to the training samples.

TensorFlow library has a working model for PixelCNN called PixelCNN++ [62]. The plus signs represent computational optimizations leaving the essential method unchanged.

The library was designed to take 2-d images with gray scale pixel values as inputs. With relatively little tinkering the algorithm was repurposed to accept 2-d Ising model samples.

E. Training and validation loss

When training neural networks there are many hyper-parameters, also known as network parameters, which are set before the model is trained. These parameters are not changed through the course of the training unlike the weight parameters which are changed using algorithms like gradient descent. While there are algorithms being used to automatically optimize the selection process of hyper-parameters, this work does not utilize those findings [63]. There are still hyper-parameters that must be tuned to avoid making obvious mistakes in the training of a model.

Figure 12 presents a common diagnostic task of plotting the training loss as a function of the epoch and the validation loss as a function of the epoch. The training loss is the value of the function that the model is attempting to reduce through gradient descent, discussed above when

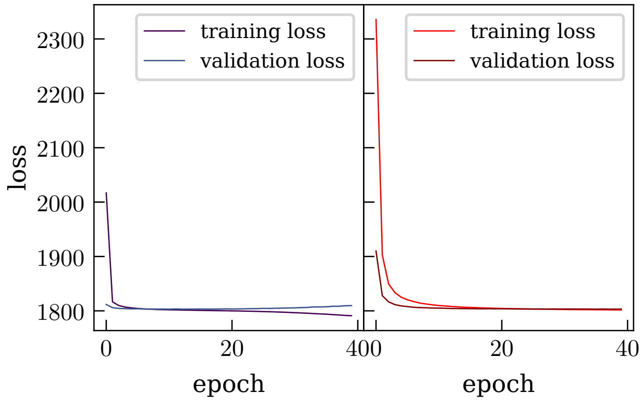

Figure 12. Showing the training loss (often just referred to as the loss) and validation loss per epoch of two separate models which differ only in the training rate. (a) Left: model with a training rate of 0.0005 which ended up being too large for the amount of epochs chosen. (b) Right: model with a lower training rate of 0.0001 which converged nicely to the test loss, showing proof the model did not over fit the data.

explaining the KL divergence. The validation loss is a metric to determine how well the network performs on new data.

Before training begins, the available data set is broken up into two groups, the training set and the validation set. The training set is shuffled and reused every epoch, acting as the ground truth through the training process. At the end of each epoch the network is shown the data from the validation set and the average loss per sample is recorded. Gradient descent is not done at this point, it is important to not train the model on the validation set in order to preserve its integrity as a diagnostic tool.

Figure 12 is plotting the loss as a function of the epoch for the same system with only a difference in the learning rate. The plot on the left shows a case where the learning rate, the magnitude at which gradient descent makes its incremental steps, is set five times higher than the network presented on the right. At around epoch number twenty, the test loss continues to decrease while the validation loss starts to increase. This is a clear sign that the model is over fitting the data. Contrasting that to the plot on the right, the lower learning rate allowed for congruent agreement between training and validation loss, showing that the model has yet to over fit. For the purposes of this work, where ample training time and computational resources are available, it is more important to not over fit the data than to hastily attempt to find convergence.

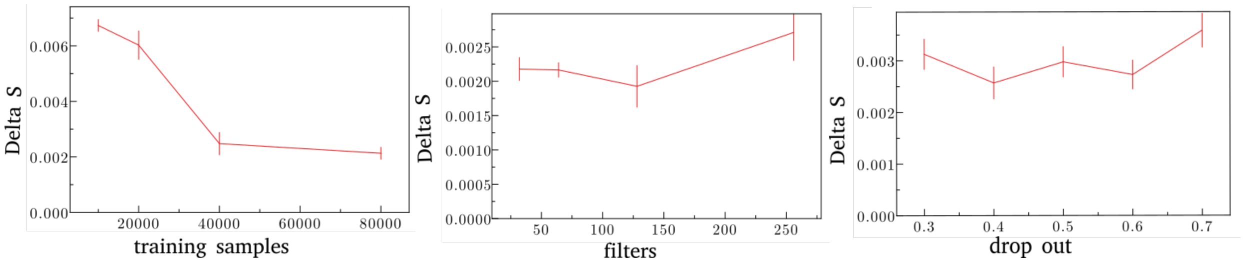

Figure 13. Explores influence of specific hyper parameters on performance. Models were trained and tested on their ability to accurately estimate the entropy of the 2-d Ising model in equilibrium at the critical temperature. Working from the left: the number of training samples used per epoch showed to have an increased performance as the amount of data was increased. Increasing the number of filters, a measure of the depth of the network (middle) had diminishing returns as the model likely began to over fit the data. Finally the drop out was increased (right), a technique of model training which randomly silences nodes in the network throughout the training process which discourages over-fitting and the value of which is the value percentage of nodes to get silenced. Increasing the drop out failed to improve model performance. It is worth noting that an increase in the drop out did increase the accuracy of the model when more filters were used.

F. Optimizing Hyperparameters

While this work is not making claims to have fully optimized the tuning of hyper parameters, a screening process was to observe their effects on the model. The simplified search was done by changing a single parameter though a given range of values and testing the effects on the final resulting entropy which that had.

In addition Keras is a framework built on top of Tensorflow 2 which has a large collection of callback functions to assist in the training of models. These proved extremely useful in that they were able to save weights only when an improvement in the training took place, in our case when the validation loss went down. For large models trained over many epochs, this helped reduce the amount of storage tremendously. Built in functions were also able to end training after a specific number of unimproved training epochs passed. This made training to saturation much simpler,

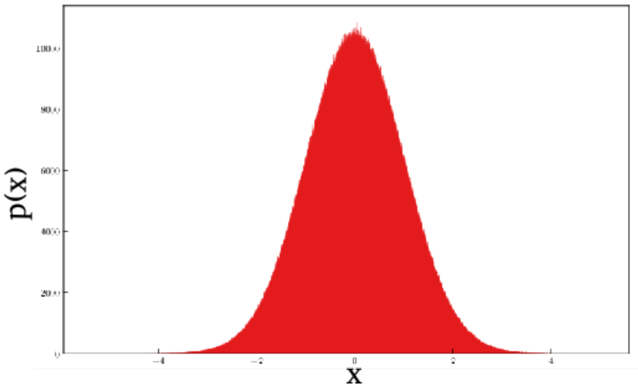

Figure 14. 2×10^7 samples generated from a Gaussian distribution by metropolis algorithm and binned into 7×10^3 bins. Because of the low dimensionality of the distribution under investigation, the histogram method works very well in building a reliable model

Results

VIII. CONTINUOUS SPACE DENSITY ESTIMATION

A. 1-Dimensional Density Estimation using histogram

A 1-dimensional test case of a Gaussian distribution was developed. Using the Metropolis-Hastings algorithm, a data set reflecting a standard Gaussian distribution was generated. The probability density function for this distribution is shown in figure 14.

After the data set was created, an appropriate range was chosen on the x-axis and the space within that range was divided into bins. Counting the number of points that fell in a given bin interval gives the bins its value, represented graphically by its height. 2×10^7 points were drawn from the above distribution are presented in figure 14 with 7×10^3 bins. It is plain to see both the familiar shape of the normal distribution while also seeing the noise caused by limited sampling.

Given the nature of the data being composed of a known number of samples, the distribution can be easily normalized. This is done by taking the output probability estimates, $p\left(x_i\right)$, for each bin and then dividing that value by the total number of samples. The entropy can then easily be calculated:

$$S \simeq -\sum_{i=1}^{n} p(x_i) \ln(p(x_i)) \Delta x, \tag{39}$$

where bin value is $[x_1, ..., x_n]$ and Δx is the size of each bin. In the limit of infinite data and infinite bins, the above equation approaches the true value for the entropy of this distribution. In such a case the summation term is written as an integral:

$$S = -\int Ae^{-\alpha x^2} \ln(Ae^{-\alpha x^2})dx \tag{40}$$

$$= -A[\ln(A)\int e^{-\alpha x^2}dx - \alpha\int e^{-\alpha x^2}x^2dx]. \tag{41}$$

The numeric approach shown above, which took under a minute of computation time on a local machine, generated samples reflecting a distribution with an entropy deviating from the original by $5.5 \times 10^{-4}\%$.

This numerical approach is a fast and reliable method for approximating the entropy for 1D distributions.

B. Combining PCA with histogram density estimation

As an example of using the factorization approach aided PCA, samples were drawn from a rotated Gaussian distribution. For this case, factorization is exact in the properly rotated basis. The distribution is defined as:

$$a = \frac{\cos(\theta)^2}{2\sigma_x^2} + \frac{\sin(\theta)^2}{2\sigma_y^2},$$

$$b = \frac{-\sin(2\theta)^2}{4\sigma_x^2} + \frac{\cos(\theta)^2}{4\sigma_y^2},$$

$$c = \frac{\sin(\theta)^2}{2\sigma_x^2} + \frac{\cos(\theta)^2}{2\sigma_y^2},$$

$$P(\mu_x, \mu_y, \sigma_x, \sigma_y, A) = Ae^{-a(x-\mu_x)^2+2b(x-\mu_x)(y-\mu_y)+c(y-\mu_y)^2}. \tag{42}$$

Here μ and σ are the means and standard deviations per axis and θ is the angle at which the rotated axis has with respect to the x-axis. A normalizes the distribution. For the simulations done in this work the mean values were set to zero and the standard deviations and magnitude were set to 1.0; θ was set to π.

This type of distribution is attractive because the analytical value of its entropy can be calculated easily. The reason for this is because the distribution is comprised of a product of two Gaussian distributions, making it completely linearly correlated.

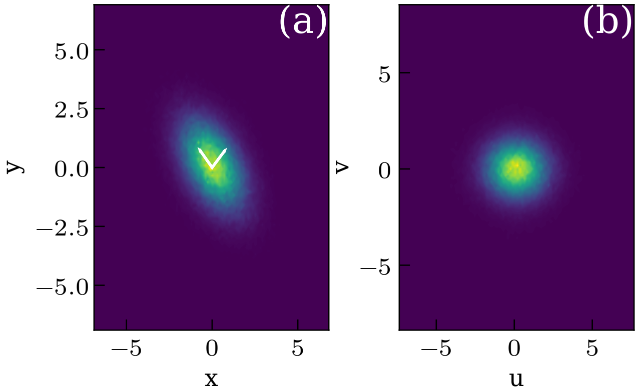

Figure 15. Rotated Gaussian distribution before (a) and after (b) PCA guided linear transformation using Monte Carlo 1,000,000 generated samples and histograms 100x100 bins. The linear correlation is almost fully erased under the new basis vectors.

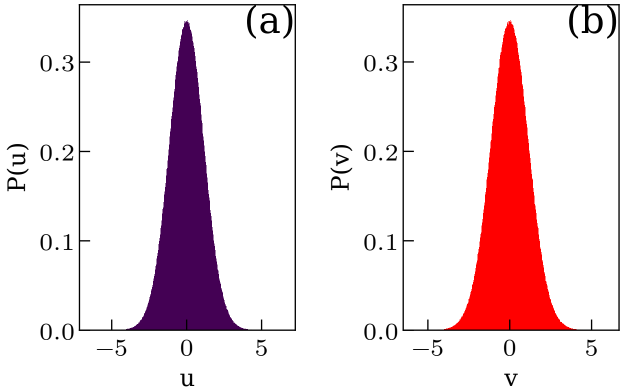

Figure 16. Normalized one dimensional data taken after PCA was done. These distributions that can be factorized and will give an accurate estimate for the entropy when added together.

In figure 15 (a) a two dimensional histogram is shown with data generated using the Metropolis-Hastings algorithm. PCA was performed and the two principle components are highlighted in white. The distribution was then transformed to coordinates where the the principle components, u and v, are its new axes and is rescaled to exhibit the symmetry seen in figure 15 (b).

From this new transformed data set the entropy of the original distribution can be calculated as though it were a product distribution of two Gaussian distributions. For this specific example, where the function with linear dependence is truly a Gaussian, we know with enough samples the system will converge perfectly with the analytically calculated entropy.

The transformed data was taken and separated along the new axes u and v. For each histogram a density was taken and added together.

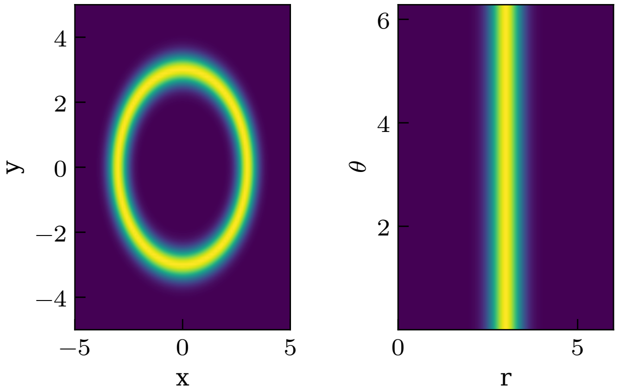

Figure 17. Example of a distribution with radial symmetry (left) and its transform to a radial coordinate system (right). PCA would fail to find meaningful principle components in the radially symmetric distribution.

C. Tricking Principle Component Analysis

Understanding the limitations of a technique is important. Some distributions exhibit Gaussian properties radially but not in Cartesian coordinates. Figure 17 (a) shows a distribution which resembles a halo and can be defined mathematically as:

$$P\left(\mu_x, \mu_y, \sigma_x, \sigma_y, A\right) = Ae^{-a(x-\mu_x)^2+c(y-\mu_y)^2}, \tag{43}$$

in Cartesian coordinates and

$$P\left(r, \theta, \sigma\right) = e^{\frac{-(r-R)}{2\sigma}} \tag{44}$$

in radial coordinates.

Performing PCA on such a distribution would be meaningless as, at the limit of infinite samples, any direction is equally likely for the first principle component. The same distribution viewed in radial coordinates (figure 17 (b)) however, can be exactly factorized even without PCA.

A more subtle example of a set of distributions that fail to be accurately described by PCA in practice are distributions that are equally balanced in correlation on multiple axes, only one of which reflects a factorizable coordinate. In figure 18 (a) we have a representation of samples drawn from such a distribution comprising four gaussians on corners of a square. A Metropolis algorithm equipped with long jumps is used here, to escape the large distance between wells. Figure 18 (b) and (c) represent two transformations that minimize correlations and would at first glance seem

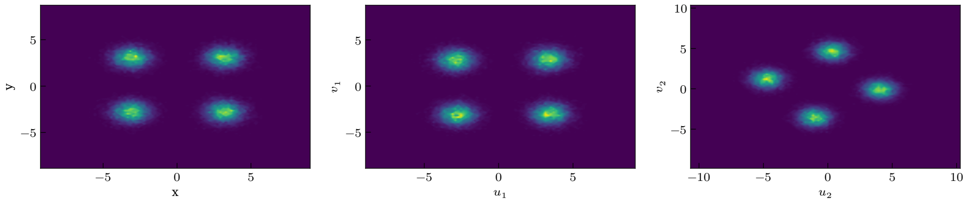

Figure 18. Examples of distributions that are highly sensitive to imbalance, rendering PCA ineffective. (a) represents 60,000 samples drawn from a distribution composed of four smaller Gaussian distributions using the Metropolis-Hastings algorithm. (b) and (c) represent the two equally likely possibilities of transformation using principle components as new basis vectors.

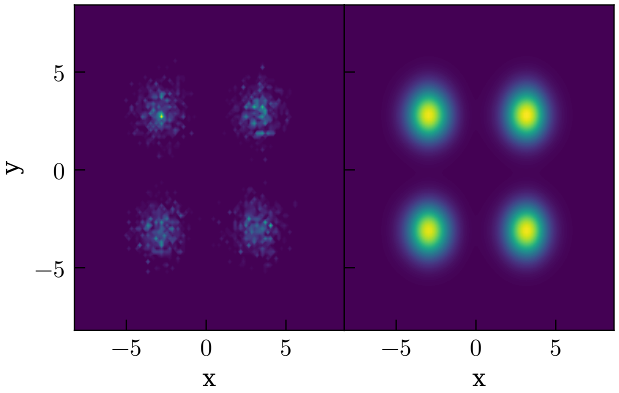

Figure 19. Example of how the distribution found in Figure 18 on page 35a estimated using a KDE.

equally likely to be chosen by PCA. Yet, it is easy to see that for finite sample sizes, the non-factorizing transformation (c) would almost always be chosen, because peaks neat the axis reduce correlation Increasing the dimensionality of the problem would allow for more areas of symmetry, further accentuating this weakness of the algorithm to accurately identify the principle components in which would allow for the reduction of dimensionality and the estimation of entropy.

D. Overcoming limited sample size

In figure 19 the same distribution which “tricked” PCA was effectively estimated using KDE. Some trial and error was needed to move through the bandwidth parameter space to find a reasonable value.

Choosing the right kernel and bandwidth is not always simple. Knowing the distribution before hand made the process trivial but in cases where a benchmark is not known, other, more complicated methods must be developed to properly investigate which parameters allow the model to

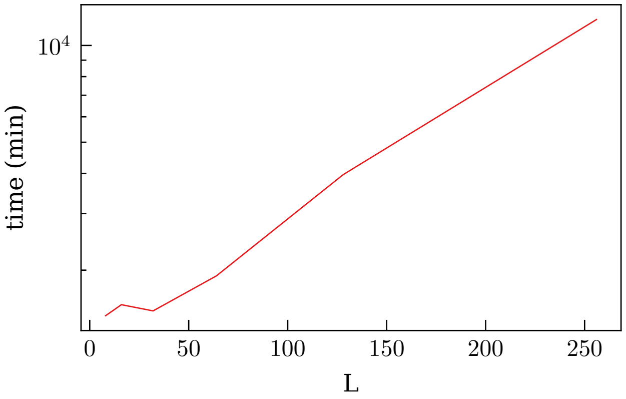

Figure 20. Training time as a function of the system size where L is a side length of a square lattice.

represent the distribution most accurately .

IX. WORKING ON HIGHER DIMENSIONAL OBJECTS

A. Analytic bench-marking of PixelCNN

The methods explored above did manage to yield satisfactory results but are either limited in accuracy (like factorization) or scale exponentially with dimension (like full histograms and KDE). A scalable method that is both flexible and robust is needed to account for the diverse and long range correlations that might exist in applications to statistical mechanics.

The PixelCNN method was used on Monte-Carlo generated samples of the Ising model at various different system sizes. The PixelCNN network was trained on those samples as its ground truth. Networks were trained on 10,000 training samples and tested on 1,000 samples. Figure 20 shows the effect that system size has on the training time of the network.

Figure 21 shows the performance of the trained network for samples in equilibrium. The network is a generative one meaning it is able to generate samples with respect to the model distribution. This capacity is illustrated in the stack of panels on the right side of figure 21. There samples generated by PixelCNN are compared to the Monte Carlo simulated samples for a range of temperatures. Before even needing to look at the analytic benchmark, it is reassuring to see that the samples generated by PixelCNN resemble those pulled from the actual distribution.

The left side of figure 21 has two panels displaying how the model performed against the analytic benchmarks. The top panel shows ΔS as a function of temperature for various different system sizes. ΔS in this case is the analytic result for the Ising model in equilibrium [64] subtracted by the entropy generated by the model.

The entropy is calculated for the model by feeding in Monte Carlo generated samples and having the model calculate the log-probability of each sample. That value is divided by the system size to give the value of the entropy per site. This was done and averaged over 1,500 samples and the standard errors were used to provide confidence intervals for the data presented in figure 21.

It should be noted that there is a spike in the error as we measure near the critical regime of the 2d Ising model which is $T_c \approx 2.67$. The PixelCNN model outperforms the MICE algorithm in acheiving a better entropy estimation as can be seen in the top left panel of figure 21. It shows both models deviation from the analytic benchmark, ΔS, and for both high temperature cases and for calculations at the critical regime PixelCNN has lower errors. Entropy around the critical temperature is the most difficult to estimate because correlations are least predictable in this regime [65]. Because of this, estimations at T_c are often used as a litmus test to the level of robustness of a method.

For larger system sizes the accuracy of the method increases. This is clearly illustrated in the bottom left panel of Figure 21 on page 38. The average error, $\langle \Delta S \rangle$, for all the calculated temperatures is plotted as a function of the system size. Monte Carlo error scales like $\frac{1}{\sqrt{N}}$ where N is the number of samples [13]. Given that the amount of information contained in an Ising model sample can be viewed as a function of the number of sites, which is L^2 in this case, an increase in accuracy should scale inversely with system size. The fact that a less steep increase is observed probably has something to do with the fact that for larger systems, more models are introduced which leaves more room for error and noise. Experiments were conducted where the model was trained on more data and the accuracy of the model increased by two orders of magnitude at the cost of comparable increases in computation time.

B. Exploring non-equilibrium phase space

Next, we consider Glauber dynamics in an oscillating magnetic field (see equation (20)). The period of oscillation, 2π, was held constant at 50 time steps per cycle. Model dynamics and its dependence on the temperature, T, and field amplitudes, H, were explored.

Figure 22 shows a collection of sampled configurations ordered in a time series which are examples of these characteristic behaviors. The initial states (top left subpanels) are random configurations. Panel (a) of figure 22 shows paramagnetic behavior, where the mean magnetization of the model follows the magnetic field. This happens every field cycle. Depending on the temperature

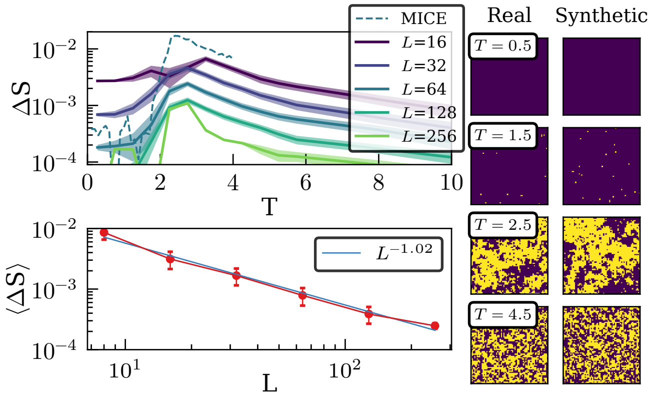

Figure 21. (a) Top left: the average error in PixelCNN entropy estimation technique as a function of system size. Portrays the accuracy of the method for several system sizes and compares the results to Ref. [36] labeled by the dashed black line. (b) Bottom left: shows the mean error over temperature in entropy estimation as a function of system sizes. (c) Right two columns: displays Monte Carlo generated samples next to samples drawn from the PDF estimated by the PixelCNN network for several different temperatures.

and the field strength, the response time of the magnetization of the model changes. Systems will either become more responsive, usually in higher temperatures, or slowly responsive and lagging behind the field, remaining saturated at a given magnetized state for most of the time. Panel (b) of figure 22 shows ferromagnetic behavior where the system becomes permanently magnetized in a direction depending on the initial value of the magnetic field. The field strength and thermal fluctuations are too low to disrupt the magnetism afterwards. Panel (c) of figure 22 shows the most interesting behavior, which arises on the border between the previous two regimes. Here the magnetic field is strong enough to influence the magnetization of the model but often not strong enough to fully flip it. In this way some long lasting correlation can begin to arise, surviving through potentially many cycles.

Having observed these behaviors by eye, order parameters to consistently and accurately define the behavior would be useful for making quantitative statements. We define the following order parameters:

$$O_{\text{ferro}} = \text{abs}\left(\text{avg}\left(M\right)\right), \tag{45}$$

$$O_{\text{para}} = \text{avg}\left(\text{abs}\left(M\right)\right) \cdot \left(1 - O_{\text{ferro}}\right), \tag{46}$$

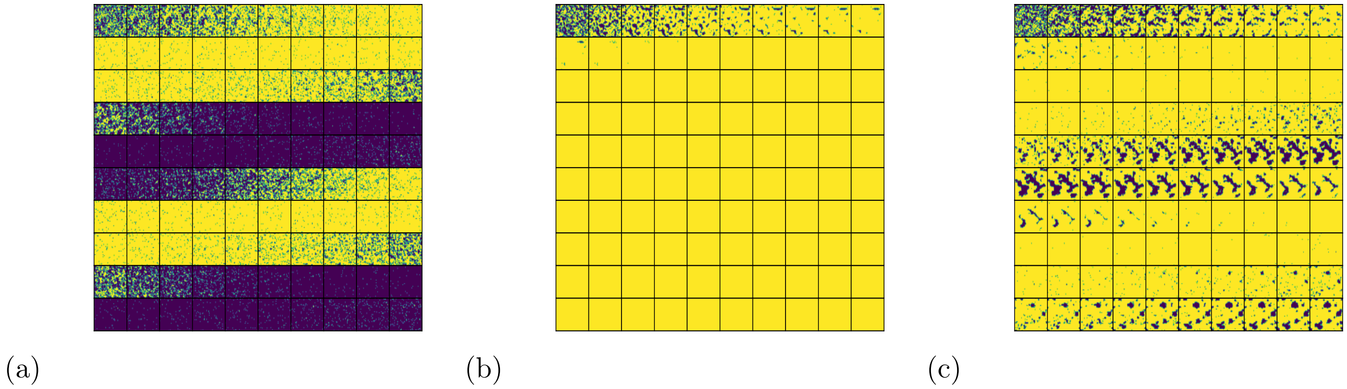

Figure 22. Characteristic behaviors which arise through investigation of the parameter space. Plots are composed of 100 frames of consecutive time steps during through a simulation of Glauber dynamics of an $L = 64$ square lattice, each under differentiating magnetic field strengths and temperatures. (a) The left most figure shows an example of dynamics within the paramagnetic regime, $H = 3$ and $T = 3$. (b) The middle plot is an example of dynamics within the ferromagnetic regime, $H = 0.5$ and $T = 0.5$. (c) The right most figure shows an example of dynamics within the chaotic regime, $H = 0.8$ and $T = 1.7$.

$$O_{\text{chaotic}} = \text{var}\left(M\right). \tag{47}$$

Where the order parameters for the ferromagnetic, paramagnetic and chaotic regime are O_{ferro} , O_{para} and O_{chaotic} respectively and M is the mean magnetization of sites for a configuration at a given time t. The averages are taken over many configurations. The variance in $O_{chaotic}$ is taken many different realizations of trajectories taken by the system.

The top three plots of figure 23 show the phase diagram constructed using the above mentioned order parameters. The level at which O_{ferro} was present is represented by a blue color, O_{para} by green and $O_{\text{chaotic}}$by red. It is worth noting that O_{chaotic} is obtained by taking the variance over many different realizations of the dynamics simulation. This order parameter also appears when the temperature is low and the magnetic field gets strong enough to cause fluctuations but not strong enough to flip the overall magnetization. The red color on the bottoms of the phase diagrams is therefore not indicative of the chaotic regime. It is also worth noting that, by our definition, the chaotic regime is larger in smaller systems where the normal phases are not sharply defined.

The lower panels of figure 23 show, for each regime, an example of the time dependence of the mean magnetization overlaid by the magnetic field. The mean magnetization, marked by the blue

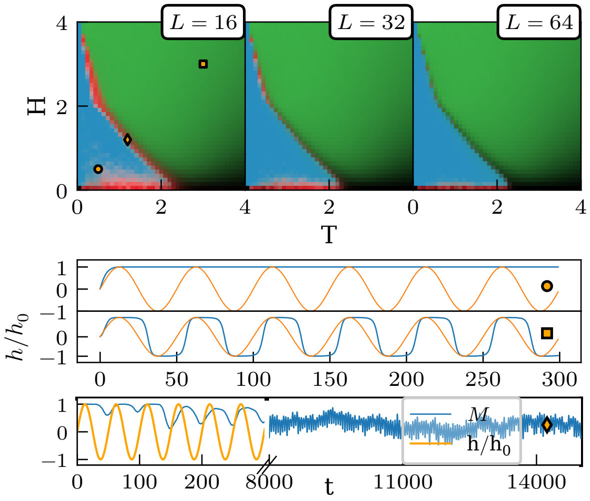

Figure 23. (a) Each pixel in the three phase diagrams (top) represents an Ising simulation run with parameters of magnetic field amplitude (H) and temperature (T). The colors show the prominence of order parameters. The three panels are system sizes $L = 16$, $L = 32$ and $L = 64$ (b) Examples of characteristic behavior of the $L = 16$ Ising model: magnetized, linear response, saturated and chaotic; going from top to bottom. Symbols denote the parameters of each example.

line, is an average value over 96 realizations of the dynamics simulation. For the case of the Chaotic regime, long time frames are presented to illustrate the varied behavior and unpredictability of the magnetization.

C. Exploring the chaotic regime

The phase diagrams were used as a reference to guide further investigations of the chaotic regime. Focusing on that region, for a given magnetic field strength, contour plots were made which depict the time averaged entropy and magnetization of the model.

Figure 24 shows the results of this investigation. It should first be clarified how the data was generated. Samples were taken for long run times over many cycles. Because of the cyclic nature of the magnetic field each cycle could be looked at as a stand alone experiment. The data was broken up by cycle and was further separated into groups numbering the same as the amount of time steps per cycle, in this case fifty. So, for example, sample 1, sample 51, sample 101 and so on, would be put into group 1. In this way all of the samples in each of the 50 groups all had the same external forces working on them. Then, a PixelCNN network was trained for each group separately. This was done for several different temperatures ranging from 0.0 to 1.0 in increments of $\Delta T = 0.1$. The resolution was increased around the area of the chaotic regime, $T = 0.3$ to $T = 0.6$, from $\Delta T = 0.1$ to $\Delta T = 0.02$. Figure 24 is therefore a contour of 22 temperatures and 50 time steps, which totals to 1,100 uniquely trained models.

In the top right corner of figure 24 the mean magnetization for $L = 32$ is plotted for all of the data sets. This metric alone does not reflect the chaotic regime as well as the estimated entropy does. The zero mean magnetization acts as a clue to the increased state of the entropy but to quantitatively learn about the entropy is a different matter entirely. Such measurements of this regime is potentially unexplored physics of an old and well worn model. The one dimensional slices at the bottom of the figure show clearly the jump in entropy during a phase transition and the low entropy of the magnetized state outside of the chaotic regime. Those can be compared to the relatively high entropy states when that same time slice is observed at a temperature in the chaotic regime.

It is of interest to observe what happens to the regime as the system size increases. In figure 25 the results for models trained on four different system sizes, $L = 8$, $L = 16$, $L = 32$ and $L = 64$, are shown. It can be seen that the chaotic regime shrinks as the system size increases, getting concentrated to a smaller range of temperatures. Due to the number of models needed to be trained to produce these figures, and the increase in training time as the input data increases, the full extent as to whether the existence of the chaotic regime is a finite size effect has not been conclusively decided one way or the other, but at the very least we expect it to be increasingly difficult to observe in larger systems.

Conclusion

Experimental data and simulations are our means of investigation to the behaviors and governing principles behind real world phenomena. For many systems the configuration space is extremely large and not tractable. For certain physical properties the full distribution is needed, one of which is the entropy. Utilizing symmetries and other assumptions of the system, it is sometimes possible to approximate the density of the system given a relatively small number of observations. However, it is often the case, that such assumptions are incorrect of unknown. In such cases, models which are both robust and flexible are of great use to theorist and sentimentalists alike.

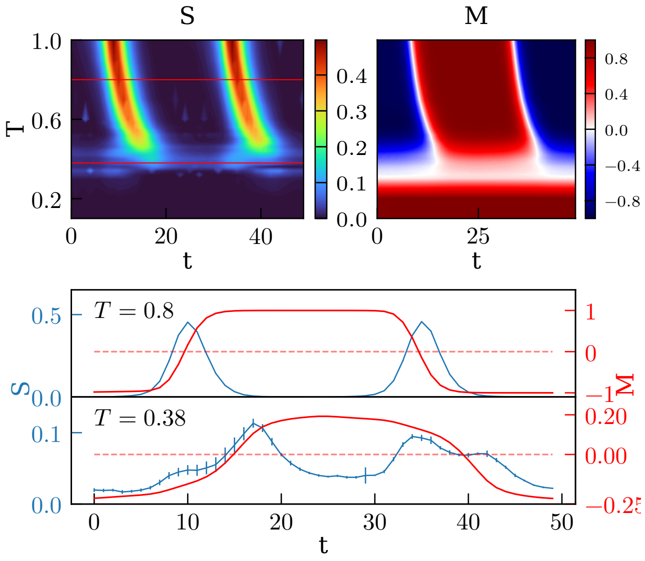

Figure 24. Describes the entropy and magnetization of the $L = 32$ Ising model under an oscillating magnet field of strength $H = 2.5$. The top left panel shows time averaged entropy calculations for temperatures around the chaotic regime. The top right panel shows time averaged magnetization calculations in the same regime. The panel below shows a high temperature region outside of the chaotic regime displaying the time averaged magnetization (red line) and entropy (blue line) over a cycle. Just below that shows measurements for a temperature within the chaotic regime, showing higher relative entropy when the effective field is strongest as the system fails to fully magnetize during that time.

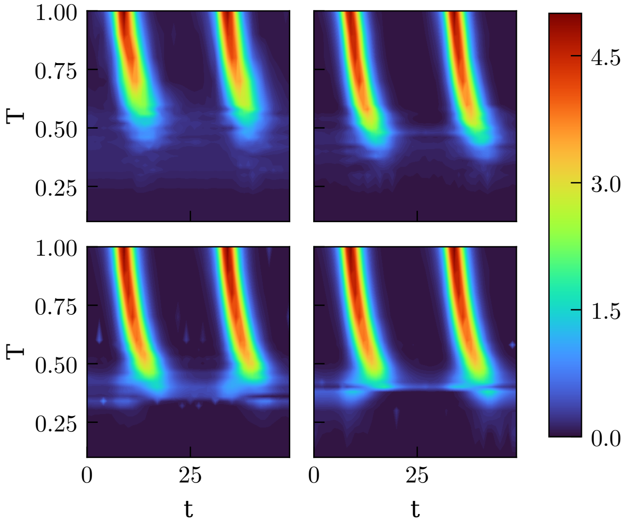

Figure 25. Shows contour plots of the PixelCNN calculated time averaged entropy for Ising models under the influence of an oscillating magnetic field, for a range of temperatures. System sizes are, from the top down and left to right, $L = 8$, $L = 16$, $L = 32$ and $L = 64$.

In this work we presented a method which makes few assumptions regarding the nature of the system and is capable of building models that are not restricted to equilibrium. The method proved viable by its accurate performance successful estimation of the entropy of large, high dimensional systems in the case of the 2d Ising model in equilibrium compared to its analytic benchmark. A numerical simulation was then carried out where the Ising model was pushed out of equilibrium by a fluctuating magnetic field and its dynamics were observed. Using simulated data and several order parameters, the systems phase space was explored, uncovering an interesting chaotic regime. Using our newly proposed method, entropy in this regime was explored.

The method implemented here was adopted from technological and methodological advancements made in the field of image processing using deep learning networks. At the time of this writing Van Der Oords initial publication on the PixelCNN algorithm has accrued over 1,900 citations in under six years. Methods like generative adversarial networks, normalizing flows and variational autoencoders are progressing concurrently and managing to model subtleties within high-dimensional distributions [66] [67] [68]. Further, methods are coming out of Google which manage to outperform all of these following models in the task of convincing image generation for full color images of size 1024x1024 [69]. These models are staggeringly large and are computed using many state-of-the-art Tensor Processing Units over the course of several days. All this is worth mentioning to show that the scope of potential statistical-mechanical systems which can be studied should in no way be limited to the 2-dimensional Ising model.

Methods of this kind and others like it, utilizing the power of NNs, should be seriously considered for exploration. The results of this work demonstrate the power of this and similar methods while the work being done outside the walls of the University prove that the surface of potential scientific exploration has only just been scratched.

Acknowledgments

I would like to start by thanking Guy Cohen for his patience and tutelage. He provided me with guidance the whole way through this process, building much of my understanding of the field from the ground up. He taught me how to approach problems and gave me the tools to solve them. I consider meeting him one of the greatest blessings of my life and certainly my career.

In addition to helping me personally, Guy also put together a fantastic group of young scientist who I can proudly say are more than just colleagues – hopefully life long friends. They helped me with research, to excel at course work and to fully enjoy life here in the city. Thanks team.

The Tel Aviv University school of Chemistry is superb in the quality of its faculty members, the courses they offer and the environment of camaraderie and support which they foster.

I would also like to thank my family for their constant support, both in the highs and lows which came to pass through the length of this degree.

[1] S. Carnot, .

[2] Ã. Brunet, T. Hocquet, and X. Leyronas, , 95.

[3] P. d. C. A. du texte Spindler, G. .-. A. du texte Meyer, and J. H. A. du texte Meerburg, “Annalen der Physik,”.

[4] R. Clausius, **2**, 1.

[5] H. Van Helmholtz, “Uber die Erhaltung der Kraft, von Dr. H. Helmholtz (1847),”.

[6] J. W. Gibbs, *Graphical Methods in the Thermodynamics of Fluids* (Connecticut Academy).

[7] “Google Books Ngram Viewer,” ().

[8] K. D. Bailey, *Social Entropy Theory*.

[9] S. Kullback and R. A. Leibler, **22**, 79.

[10] A. Jakimowicz, **22**, 452.

[11] C. Bourke, J. M. Hitchcock, and N. V. Vinodchandran, **349**, 392.

[12] W. T. Kelvin, J. Larmor, and J. P. Joule, *Mathematical and Physical Papers* (Cambridge, University Press).

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing* (Cambridge University Press) 1aAOdzK3FegC.

[14] J. Lee, **71**, 211.

[15] B. J. Alder and T. E. Wainwright, **27**, 1208.

[16] M. E. J. Newman and G. T. Barkema, *Monte Carlo Methods in Statistical Physics* (Oxford University Press).

[17] J. M. Rickman and D. J. Srolovitz, **99**, 7993.

[18] P. M. C. de Oliveira, T. J. P. Penna, and H. J. Herrmann, “Broad Histogram Method,” arXiv:cond-mat/9610041.

[19] A. M. Ferrenberg and R. H. Swendsen, , 5.

[20] J.-S. Wang, **127**, 10.1016/S0010-4655(00)00016-3.

[21] F. Wang and D. Landau, **86**, 2050.

[22] M. Souaille and B. Roux, **135**, 40.

[23] C. Zhou and R. N. Bhatt, **72**, 025701.

[24] E. T. Jaynes, **106**, 620.

[25] D. Frenkel and A. J. C. Ladd, **81**, 3188.

[26] C. P. Herrero and R. Ramirez, **568–569**, 70, arXiv:1307.3950.

[27] C. Peter, C. Oostenbrink, A. van Dorp, and W. F. van Gunsteren, **120**, 2652.

[28] T. S. Komatsu, N. Nakagawa, S.-i. Sasa, and H. Tasaki, **159**, 1237, arXiv:1405.0697.

[29] L. Ferreira Calazans and R. Dickman, **99**, 032137.

[30] M. Kastner and M. Promberger, , cond.

[31] G. Darbellay and I. Vajda, **45**, 1315.

[32] S.-k. Ma, , 20.

[33] R. Avinery, M. Kornreich, and R. Beck, **123**, 178102, arXiv:1709.10164.

[34] A. Bulinski and A. Kozhevin, "Statistical Estimation of Conditional Shannon Entropy," arXiv:1804.08741 [math, stat].

[35] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, and R. D. Hjelm, "MINE: Mutual Information Neural Estimation," arXiv:1801.04062 [cs, stat].

[36] A. Nir, E. Sela, R. Beck, and Y. Bar-Sinai, **117**, 30234, 33214150.

[37] P. S. Laplace, *ThÃ©orie analytique des probabilitÃ©s;* (Paris, Ve. Courcier).

[38] T. M. Cover and J. A. Thomas, .

[39] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, **21**, 1087.

[40] W. K. Hastings, **57**, 97.

[41] S.-K. Au and J. L. Beck, **16**, 263.

[42] "6. Phase Transitions: Introduction to Statistical Mechanics," ().

[43] P. Curie, *PropriÃ©tÃ©s magnÃ©tiques des corps Ã diverses tempÃ©ratures* (Gauthier-Villars et fils) QhMywOm$_y$*NsC.L. Onsager*, **65**, 117.

[45] N. D. H. Dass, B. E. Hanlon, and T. Yukawa, **368**, 55, arXiv:hep-th/9505076.

[46] A. E. Ferdinand and M. E. Fisher, **185**, 832.

[47] U. Wolff, **62**, 361.

[48] R. J. Glauber, **4**, 294.

[49] H. Hotelling, **24**, 417.

[50] Drleft, “English: Comparison of a histogram and a kernel density estimate.” (28 September 2010, 12:52 (UTC)).

[51] C. E. Shannon, , 55.

[52] A. N. Kolmogorov, **25**, 369, 25049284.

[53] J. Ziv and A. Lempel, **24**, 530 ().

[54] J. Ziv and A. Lempel, **23**, 337 ().

[55] I. M. Pu, *Fundamental Data Compression* (Butterworth-Heinemann) Nyt0HgC81I4C.

[56] D. Benedetto, E. Caglioti, and V. Loreto, **88**, 048702.

[57] J. Jiao, H. H. Permuter, L. Zhao, Y.-H. Kim, and T. Weissman, **59**, 6220, arXiv:1201.2334.

[58] G. U. Yule, **84**, 497, 2341101.

[59] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel Recurrent Neural Networks,” (), arXiv:1601.06759 [cs].

[60] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, “Conditional Image Generation with PixelCNN Decoders,” (), arXiv:1606.05328 [cs].

[61] M. Germain, K. Gregor, I. Murray, and H. Larochelle, “MADE: Masked Autoencoder for Distribution Estimation,” arXiv:1502.03509 [cs, stat].

[62] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, .

[63] J. Bergstra, R. Bardenet, Y. Bengio, and B. KÃ©gl, in *Advances in Neural Information Processing Systems*, Vol. 24 (Curran Associates, Inc.).

[64] P. D. Beale, **76**, 78.

[65] I. O. Morales, E. Landa, C. C. Angeles, J. C. Toledo, A. L. Rivera, J. M. Temis, and A. Frank, **10**, e0130751.

[66] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, “Normalizing Flows for Probabilistic Modeling and Inference,” arXiv:1912.02762 [cs, stat].

[67] A. Vahdat and J. Kautz, “NVAE: A Deep Hierarchical Variational Autoencoder,” arXiv:2007.03898 [cs, stat].

[68] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive Growing of GANs for Improved Quality, Stability, and Variation,” arXiv:1710.10196 [cs, stat].

[69] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, , 1.